# Subsumption Architectures

Natural language and ARtificial intelligence Group

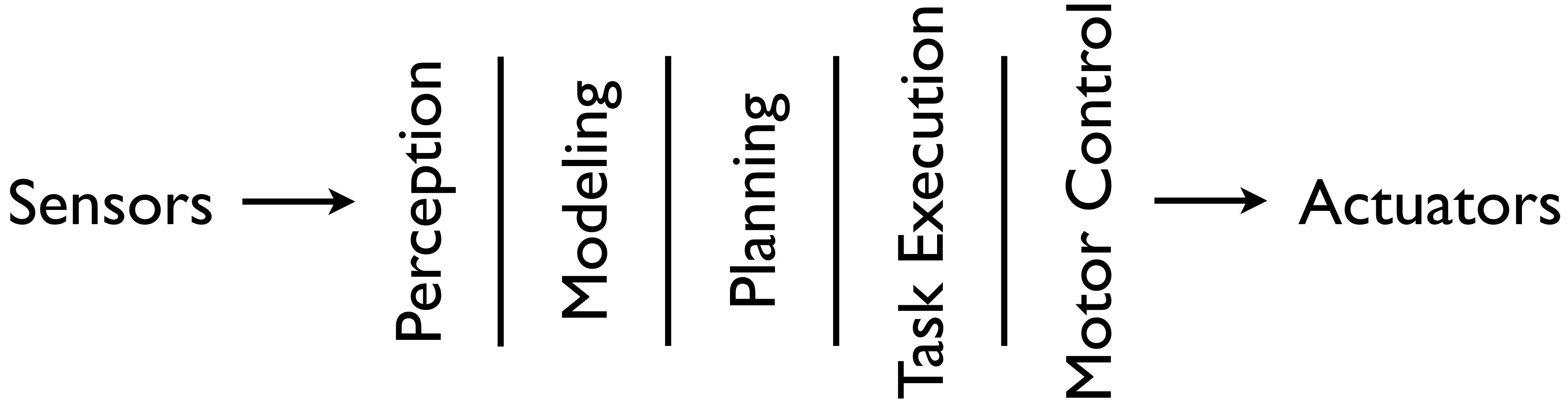
# Reactive Agent Paradigm

- Second winter brought radical shifts in AI
- This paradigm was born from robotics
- The leader of this shift was Rod Brooks
- Introduced Grounding Theory and Subsumption Architectures

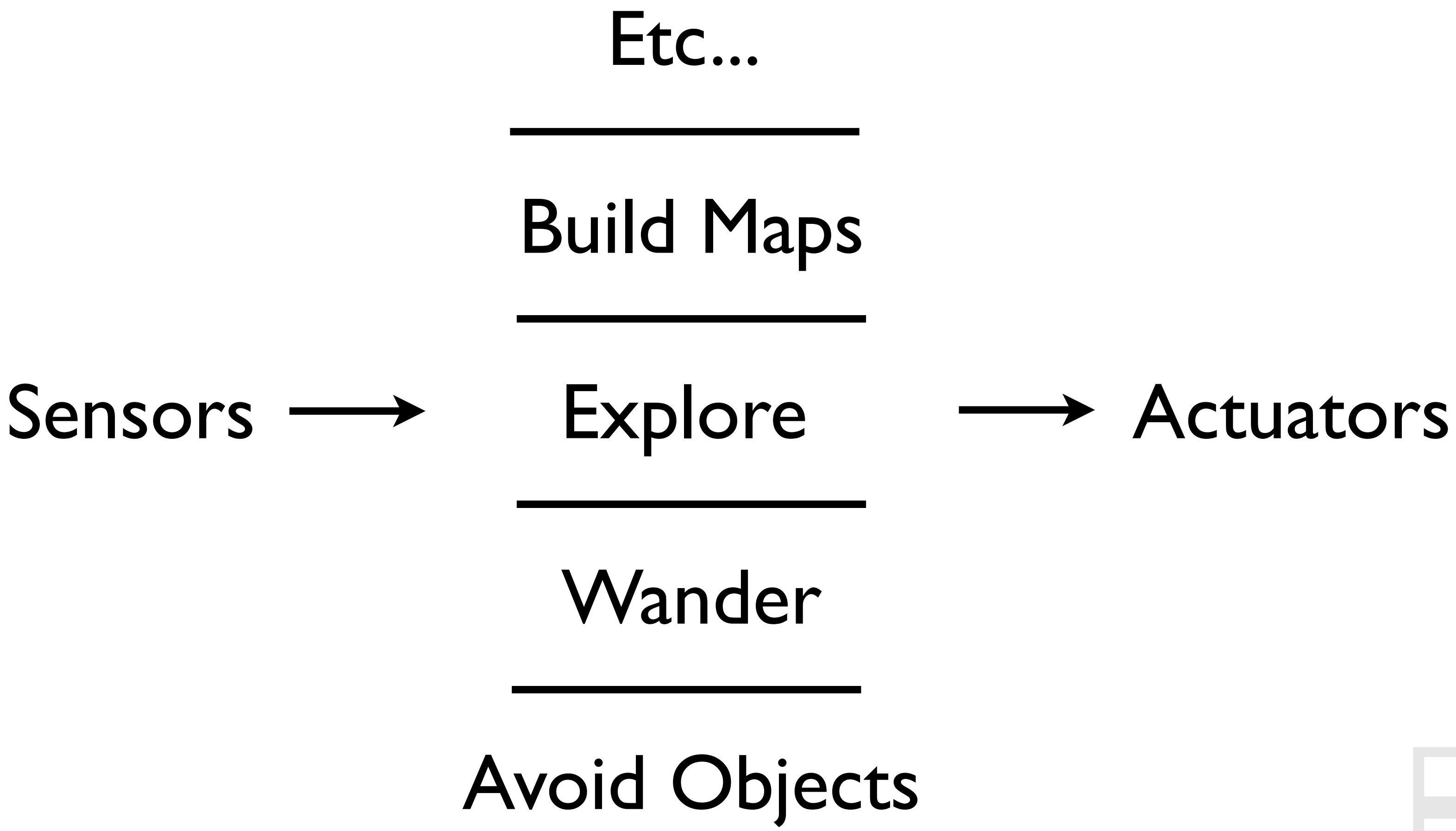
# Grounding Theory

- "[...] to build a system that is intelligent it is necessary to have its representations grounded in the physical world."
- "[...] the world is its own best model."
- "[...] and must extract all its knowledge from physical sensors."
- "[...] complex behavior may be a product of an extremely complex environment."

# Traditional Sense-Plan-Act Loop



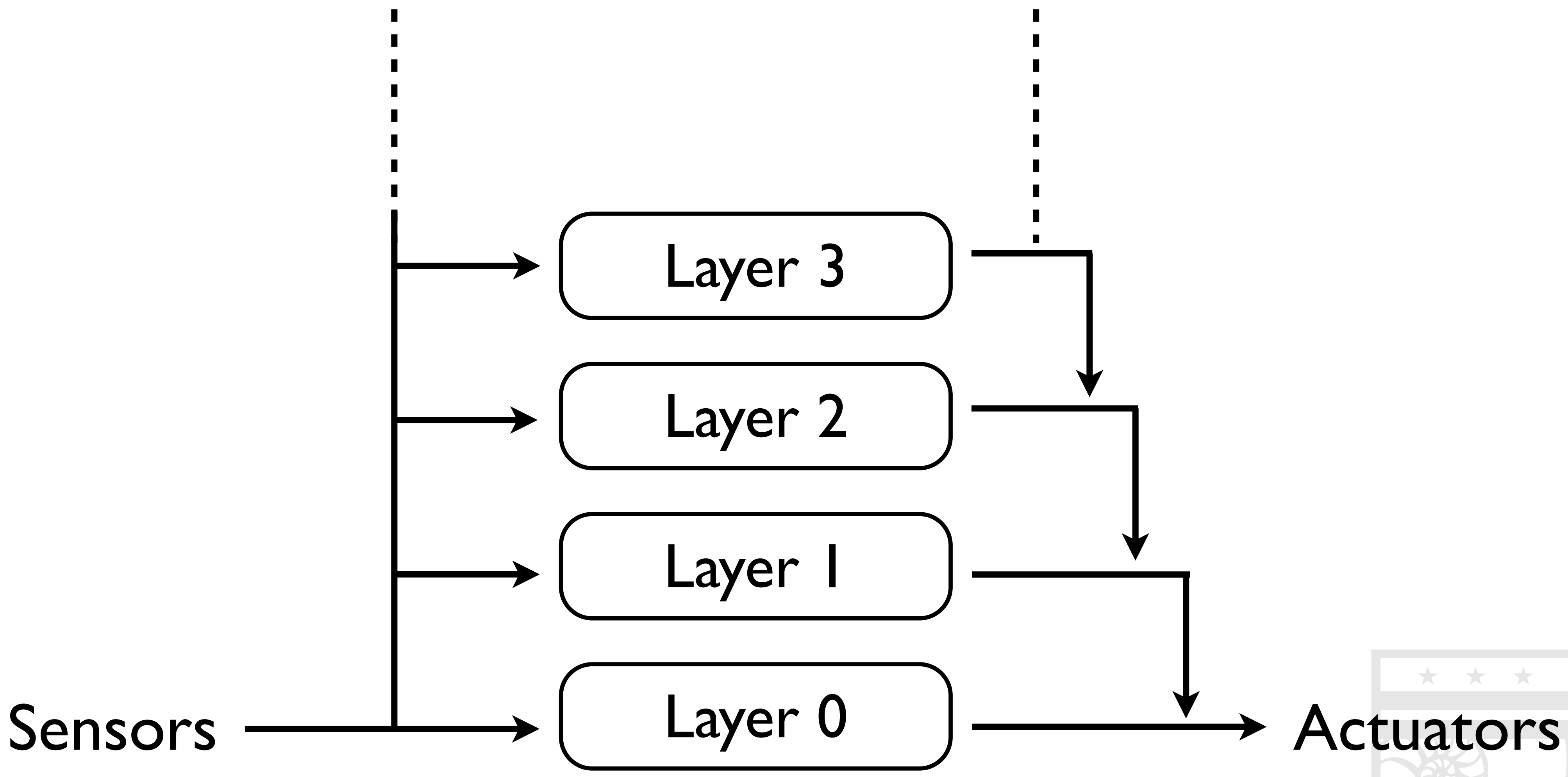
# Subsumption Architectures



## Subsumption Architecture Cont.

- Behavior is broken up into many task oriented behaviors (called modules)
- Modules are hierarchically linked
- Lower level modules can inhibit higher level modules execution

# Subsumption Architectures Cont.



## Example Time

- Modules are needed to create the behaviors
- Can be created with classes or functions

```
class SAModule (object):
    def __init__(self, func):
        self.run = func
```

# Example Continued

- Functions supplied to modules should follow some convention for inhibiting behaviors

```
def sample_func(bot, sense_info):
    """
    Returns true to inhibit any
     higher level functions.
    """
    if sense_info["at_edge"]:
        # Do some stuff
        return True
    else:
        # Do other stuff or just:
        return False
```

# Example Continued

- Inhibition can be achieved using a simple for loop that breaks

```
modules = [level_0, level_1, level_2]

for module in modules:
    if module.run()
        break
```

# Subsumption in CTF

- Class methods will serve as modules
- __init__ method will create ordering
- Iterate method will execute the architecture
- Uniform sensory-info object is necessary

# Subsumption in CTF

- Initialize the module order

```
class TeamBrooks (CTFPlayer):
    def __init__(self):
        CTFPlayer.__init__(self)
        self.modules = [self.level0,
                        self.level1,
                        self.level2]
```

## Subsumption in CTF

- Define avoid edge behavior

```
def level0(self, sensor_info):
    """
    Avoid the edge of the map.
    """
    if sensor_info["at_edge"]:
        self.setSpeed(1)
        self.turnRight()
        return True
    else:
        return False
```

# Subsumption in CTF

- Define find enemy territory behavior

```
def level1(self, sensor_info):
    """
    Head toward the enemy side.
    """
    if sensor_info["on_my_side"]:
        angle = self.getAngle(self.getOtherHomeLocation())
        self.setSpeed(1)
        if angle < 0:
            self.turnLeft()
        else:
            self.turnRight()
        return True
    else:
        return False
```

# Subsumption in CTF

- Define flag locating behavior

```
def level2(self, sensor_info):
    """
    Find the opponents flag.
    """
    flag = sensor_info["opponent_flag"]
    if sensor_info["on_other_side"] and flag:
        angle = self.getAngle(flag.getLocation())
        self.setSpeed(1)
        if angle < 0:
            self.turnLeft()
        else:
            self.turnRight()
        return True
    else:
        return False
```

# Subsumption in CTF

- Create a sensory-information object

```
def make_sensor_object(self):
    """
    Package sensory information into a nice bundle.
    """
    sensor_info = {}
    if self.getMyHomeLocation() == self.getLocation():
        sensor_info["on_other_side"] = False
        sensor_info["on_my_side"] = True
    else:
        sensor_info["on_other_side"] = True
        sensor_info["on_my_side"] = False
    sensor_info["opponent_flag"] =
        self.senseOtherFlag() or False
    sensor_info["at_edge"] = self.detectEdge()
    return sensor_info
```

# Subsumption in CTF

- Run it

```
def iterate(self):
    """
    Run the subsumption architecture
    """
    sensor_info = self.make_sensor_object()
    for module in self.modules:
        if module(sensor_info):
            break
    CTFPlayer.iterate(self)
```

## Lets see it in action!

- 57 lines of actual code
- 72 with comments
- Smart(ish) behaviors